

目次

1	一般的な基礎	11
1.1	リアルタイム	11
1.2	フェーズ駆動プロセスモデル：V モデル	11
1.3	ビルドプロセス：モデルから実行可能ファイルまで	12
1.4	デバッグレビューの説明	22
1.5	まとめ	23
2	マイクロコントローラ技術の基礎	24
2.1	マイクロコントローラの構造	24
2.2	コード処理	26
2.3	(メモリ) アドレッシング、アドレッシングモード	29
2.4	ウェイトステート、バーストアクセス	34
2.5	キャッシュ	36
2.6	パイプライン	40
2.7	割り込み	41
2.8	トラップ/例外	42
2.9	データ整合性	42
2.10	デスクトッププロセッサと組込みプロセッサの比較	44
2.11	まとめ	45
3	オペレーティングシステム	47
3.1	OS なし：無限ループと割り込み	47
3.2	OSEK/VDX	50
3.3	協力型マルチタスキングと抑止型マルチタスキング	55
3.4	POSIX	62
3.5	まとめ	67
4	時間論	68
4.1	タイミングパラメータ	68
4.2	確率的側面	74
4.3	CPU 負荷	77
4.4	バス負荷	86
4.5	論理実行時間 (LET)	87
4.6	まとめ	88
5	タイミング解析技術	89
5.1	概要、レベル分け	89
5.2	用語解説	92
5.3	静的コード解析	93
5.4	コードシミュレーション	102
5.5	実行時間測定	114
5.6	ハードウェアベーストレーシング	125
5.7	ソフトウェアの計測に基づくトレース	144
5.8	スケジューリングシミュレーション	160
5.9	静的スケジューリング解析	168
5.10	最適化のための進化的アルゴリズム	180
5.11	V モデルにおけるタイミング解析技術の配置	183

6	タイミング問題の実践例	185
6.1	すべての割り込みはどこから来るのでしょうか。	185
6.2	OSEK ECC：ベストな選択とは言い難い	186
6.3	リセットから 17 分後に稀にクラッシュが発生します	189
6.4	センサデータの欠落または二重受信	190
6.5	ハンドブレーキをかけた状態のレース中	195
6.6	測定は静的コード解析よりも大きな WCET を提供します	196
6.7	ネットワーク管理メッセージが早すぎる場合があります	197
6.8	シリーズプロジェクトにおけるシームレスなタイミングプロセス	199
6.9	タイミング解析で自動車メーカーが 1200 万ユーロを節約	199
6.10	まとめ	200
7	マルチコア、メニーコア、マルチ ECU のタイミング	201
7.1	マルチコアの基礎	201
7.2	さまざまなタイプの並列実行	207
7.3	データ整合性、スピンのロック	214
7.4	メモリアドレスのクロウニング	219
7.5	まとめ	222
8	実行時間の最適化	224
8.1	スケジューリングレベルでの実行時間の最適化	224
8.2	実行時間が最適化されたメモリ使用量	228
8.3	コードレベルでの実行時間最適化	233
8.4	実行時間の最適化のまとめと「ロードマップ」	251
9	開発プロセスにおける方法論	254
9.1	タイミングリファレンスの要件	254
9.2	プロジェクトパートナー間の協力	262
9.3	タイミングコンセプト、スケジューリングレイアウト、OS コンフィギュレーション	263
9.4	実行時間デバッグ	264
9.5	実行時間の最適化	264
9.6	実行時間保護	265
9.7	将来の機能の早期検討	267
9.8	シリーズのタイミング保護	268
9.9	ポジティブな例：CoReMa Vitesco Technologies	270
9.10	まとめ	271
10	AUTOSAR	273
10.1	AUTOSAR Classical プラットフォーム (CP)	274
10.2	AUTOSAR Adaptive プラットフォーム (AP)	277
10.3	TIMEX (AUTOSAR タイミング拡張機能)	286
10.4	ARTI (AUTOSAR/ASAM 実行時インタフェース)	288
10.5	テクニカルレポート「タイミング解析」	294
10.6	まとめ	294
11	セーフティ、ISO 26262	296
11.1	基礎	296
11.2	安全規格とタイミング – タイミング保護	298

11.3	タイミング保護ツール	300
11.4	法的側面	300
11.5	まとめ	301
12	展望	302